HiQLab: Simulation of Resonant MEMS

David Bindel
UC Berkeley, CS Division
Overview

- Background, target applications, and grand vision
- Simple “from scratch” examples
- Anchor loss in a disk resonator
- Thermoelastic damping in a beam
- Model reduction of a checkerboard resonator
- Current status and planned work
- Q & A and feedback
Application: RF MEMS

- Systems of MHz-GHz mechanical resonators
- Useful for
  - Frequency references
  - Filter elements
  - Sensing elements
Want to minimize damping

- Electronic filters have too much
- Understanding of damping in MEMS resonators is lacking

Characterize damping by $Q$

- Non-dimensionalized damping in a one-variable system:

$$\frac{d^2 u}{dt^2} + Q^{-1} \frac{du}{dt} + u = F(t)$$

- For a resonant mode with frequency $\omega \in \mathbb{C}$:

$$Q := \frac{\Re(\omega)}{2\Im(\omega)} = \frac{\text{Stored energy}}{\text{Energy loss per radian}}$$

Goal: Make $Q$ big
Sources of damping

- Fluid damping
  - Air is a viscous fluid ($Re \ll 1$)
  - Can operate in a vacuum
  - Shown not to dominate in many RF designs

- Anchor loss
  - Elastic waves radiate from structure

- Thermoelastic damping
  - Volume changes induce temperature change
  - Diffusion of heat leads to mechanical loss

- Material losses (catch-all)
  - Low intrinsic losses in silicon, diamond, germanium, etc.
  - Greater material losses in metals
What makes it hard?

- Physical issues
  - Not clear which loss mechanisms dominate
  - Need physically realistic models
  - Need good material property estimates

- Mathematical issues
  - Operating modes are in the middle of the spectrum
  - Damping makes the problem naturally nonsymmetric
  - \( \omega \) must be accurately resolved to estimate \( Q \)
  - Symmetries and near-symmetries lead to degeneracy
  - Dependence on parameters is critical
Goal for HiQLab

- Goal is to understand resonant MEMS behavior
  - Initially targeting high-frequency MEMS
  - Particularly care about damping
  - Want to experimentally verify our simulations

- Reason for the simulator
  - Develop physically realistic models
  - Develop fast solvers
  - Help designers
Software structure

- Element models (C++)
- Mesh management and system assembly (C++)
- Solvers
  - Standard low-level solver libraries (Fortran, C/C++)
  - High-level solvers provided by MATLAB
  - Specialized algorithms in MATLAB or C++
- Mesh description (Lua)
- User interfaces (MATLAB, Lua)
Software used

“Lesser artists borrow. Great artists steal.”
– Picasso, Dali, Stravinsky?

- Lua: www.lua.org
  ♦ Evolved from simulator data languages (DEL and SOL)
  ♦ Pascal-like syntax fits on one page; complete language description is 21 pages
  ♦ Fast, freely available, widely used in game design
- MATLAB: www.mathworks.com
  ♦ “The Language of Technical Computing”
  ♦ Good sparse matrix support
  ♦ Star-P: http://www.interactivesupercomputing.com/
- Standard numerical libraries: ARPACK, UMFPACK
- Describe a simple cantilever
- Compute and plot the modes of vibration
- Perform a parameter study
require '../common.lua'

l = 10e-6 -- Beam length
w = 2e-6  -- Beam width
dense = 0.5e-6 -- Approximate element size
order = 2   -- Order of elements
nen   = 9   -- Number of element nodes

- Common header file defines materials, block generator
- Define symbols for geometry and meshing parameters
- “–” indicates the start of a comment
Describing the mesh

mesh = Mesh:new(2, nen, 2)
mat = make_material('silicon2', 'planestrain', order)
blocks( { 0, 1 }, { -w/2.0, w/2.0 }, mat )

- Define a new mesh (which *must* be called mesh)
  - Number of dimensions = 2
  - Number of element nodes = nen
  - Number of unknowns per node = 2
- Define a polysilicon material in plane strain
- Mesh the region \([0, l] \times [-w/2, w/2]\)
  - Element size \(h\) is determined by dense
Describing BCs

```lua
mesh:set_bc(function(x,y)
  if x == 0 then return 'uu', 0, 0; end
end)
```

- Define boundary conditions with a function evaluated at every node
- Function returns
  - A string to specify displacement or force BCs
  - Values of boundary displacements or forces
- Examples:
  - Returns nothing – no boundary conditions
  - 'uu', 0, 0 – zero displacement in both \( x \) and \( y \)
  - 'u', 0 – zero displacement in \( y \) only
  - 'f', 1 – unit force in the \( x \) direction
  - 'uf', 0, 1 – zero \( x \) displacement, unit force in \( y \)
Starting in MATLAB

>> run ../../init.m

HiQlab 0.1

Copyright : Regents of the University of California
Build system: i686-pc-linux-gnu
Build date : Wed Dec 15 12:50:48 PST 2004
Bug reports : dbindel@cs.berkeley.edu

- In MATLAB, you must run init first
- If you don’t see the banner, there is a problem
Loading and plotting

```matlab
>> mesh = Mesh_load('beammesh.lua');
>> plotmesh(mesh); axis equal
```
Computing modes

\[(K - \omega^2 M)v = 0\]

\[
\begin{align*}
\text{>> } [M,K] &= \text{Mesh\_assemble\_mk(mesh)}; \\
\text{>> } [V,D] &= \text{eigs(K,M, 5, 'sm')}; \\
\text{>> } w &= \text{sqrt(diag(D))/2/pi};
\end{align*}
\]

\[
w =
\begin{align*}
1.0e+08 \times \\
0.2791 + 0.0000i \\
1.4947 - 0.0000i \\
2.2295 - 0.0000i \\
3.5475 - 0.0000i \\
5.8811 - 0.0000i
\end{align*}
\]
>> opt.axequal = 1;
>> opt.deform = 1;
>> Mesh_scale_u(mesh, V(:,1), 2, 1e-6);
>> plotfield2d(mesh, opt)
Parameter sweep

Change one line in Lua file:

\[ l = l \text{ or } 10\text{e-6} \quad \text{-- Beam length} \]

Now set \( l \) from MATLAB:

\[ l = \text{linspace}(10\text{e-6}, 20\text{e-6}, 11); \]
\[ \text{for } k = 1:\text{length}(l) \]
\[ \quad \text{param.l} = l(k); \]
\[ \quad \text{mesh} = \text{Mesh_load}('beammesh2.lua', \text{param}); \]
\[ \quad [\text{M}, \text{K}] = \text{Mesh_assemble_mk}(	ext{mesh}); \]
\[ \quad w(k) = \sqrt{\text{eigs}([\text{K}, \text{M}, \text{l}, \text{'sm'}])/2/\pi}; \]
\[ \quad \text{Mesh_delete}(	ext{mesh}); \]
\text{end}
\[ \text{plot}(l, \text{real}(w)); \text{axis tight} \]
- Change two lines to plot *several* frequencies
- Notice crossing behavior of third and fourth modes
Running from Lua

[tutorial]: hiqlab driver.lua

1 : 27.908006867444 MHz
2 : 149.46698798698 MHz
3 : 222.95156567429 MHz
4 : 354.75135363746 MHz
5 : 588.11440349498 MHz

- Can run simulations in standalone code
- Missing some capabilities (esp graphics)
- Less overhead than MATLAB interface
Lua driver file

beamf = loadfile 'beammesh.lua'

w0 = 0 -- Frequency estimate
nev = 5 -- Number of eigs
ncv = 10 -- Size of space (~2 nev)
dr = {} -- Real parts of eigs
di = {} -- Imag parts of eigs

-- Load mesh and compute eigs
beamf()
mesh:initialize()
compute_eigs(mesh, w0, nev, ncv, dr, di);

-- Print eigs
for k = 1,5 do
    print(k, ':', dr[k]/2e6/pi, 'MHz')
end
SiGe disk resonators built by E. Quévy
Axisymmetric model with bicubic mesh, about 10K nodal points
Substrate model

Goal: Understand energy loss in this resonator
- Dominant loss is elastic radiation from anchor.
- Disk resonator is much smaller than substrate
- Very little energy leaving the post is reflected back
  - Substrate is semi-infinite from disk's perspective

Possible semi-infinite models
- Matched asymptotic modes
- Dirichlet-to-Neumann maps
- Boundary dampers
- Perfectly matched layers
Perfectly matched layers

- Apply a complex coordinate transformation
- Generates a non-physical absorbing layer
- No impedance mismatch between the computational domain and the absorbing layer
- Idea works with general linear wave equations
  † First applied to Maxwell’s equations (Berengér 95)
  † Similar idea introduced earlier in quantum mechanics (exterior complex scaling, Simon 79)
Scalar wave example

Clamp solution at transformed end to isolate outgoing wave.
Defining the PML

- Only need $dz/dx = 1 - i\sigma(x)$
- Usually choose $\sigma$ to be piecewise linear
  - $\sigma = 0$ on ordinary domain
  - $\sigma > 0$ in PML region
- In higher dim, transform each $x_i$ independently
Scalar wave in HiQLab

```lua
require '../common.lua'

Ne1 = Ne1 or 10 -- Elements in first region
Ne2 = Ne2 or 10 -- Elements in second region
order = order or 3 -- Polynomial order
nen = order+1; -- Number of element nodes

mesh = Mesh:new(1,nen,1);
D = mesh:own( PMLScalar1d:new(1, 1, nen) );
mesh:add_block(-Ne1, Ne2, order*(Ne1+Ne2)+1, D, order);
```

- Mesh \([-N_1, N_2]\) with cubic elements
- Define a material for a scalar wave with PML
Scalar wave BCs

epw = epw or 10 -- Elements per wave
dpw = dpw or 1  -- Damping per wave

D:set_stretch(function(x)
    return max(x*dpw/epw,0)
end)

mesh:set_bc(function(x)
    if x == -Ne1 then return 'u', 1 end
end)

- \( \sigma(x) = 0 \) on \([-N_1,0]\)
- \( \sigma(x) \) varies linearly from 0 to 1 on \([0,N_2]\)
  - \( \sigma_{\text{max}} \) small only for demonstration
  - A more reasonable \( \sigma_{\text{max}} \) would be 10 – 40
- Displacement BCs at left end generate waves
```matlab
mesh = Mesh_load('pml1d.lua');
k = 2*pi/10;
Mesh_make_harmonic(mesh, k);
[M, K] = Mesh_assemble_mk(mesh);
F = Mesh_assemble_R(mesh);
u = -(K - k^2 * M) \ F;
Mesh_set_u(mesh, u);
```

- Mesh_make_harmonic sets $v = i\omega u$ and $a = -\omega^2 u$
- $F$ is the forcing vector from BCs
- $u$ is the time-harmonic response
x = Mesh_get_x(mesh);
u = Mesh_get_disp(mesh);
plot(x, real(u), x, imag(u));
Mesh_delete(mesh);
Disk mesh

Mesh of mapped blocks. Arguments specify block size, type of element, and \((x, y)\) coordinates of each corner.

```
mesh:add_block_shape(npostx, ndisky, pelt, order, [0, 0, 0, hpost, rpost2, 0, rpost, hpost])
```
Disk mesh

Mesh:tie(dense/100)

Mapped blocks are tied together to form disk mesh. Usually use higher mesh density than shown.
Forced response

```matlab
mesh = Mesh_load('diskmesh.lua', param);
[M,K] = Mesh_assemble_mk(mesh);
F = Mesh_assemble_R(mesh);
u = -(K - wforce^2*M) \ F;
Mesh_scale_u(mesh, u, 1, 1e-6);
plotcycle2d(mesh,1,plotopt);
```
Time-averaged energy flux

\[
\bar{E} = T^{-1} \int_0^T \sigma(t) \cdot v(t) \, dt
\]

- Compute \( \bar{E} = T^{-1} \int_0^T \sigma(t) \cdot v(t) \, dt \)
- Energy flows are strongest at surfaces
- Stress singularity at corner sprays energy

```matlab
Mesh_make_harmonic(mesh, wforce);
p = Mesh_get_x(mesh);
E = Mesh_mean_power(mesh);
quiver(p(1,:), p(2,:), E(1,:), E(2,:));
```
In a loop, compute

```matlab
param.hdisk = t(k);
mesh = Mesh_load('diskmesh.lua', param);
[M,K] = Mesh_assemble_mk(mesh);
[VV,w,Q] = pml_mode(M,K,w0,2);
Mesh_delete(mesh);
```
- Surprising variation in $Q$ as film thickness changes
- Confirmed by experiment on a set of 40 $\mu m$ disks
- Effect comes from interaction of radial and bending modes
Truth in advertising

Data from a set of 30µm radius disks.
Thermoelastic damping (TED)

\[ \begin{align*}
\sigma &= C \varepsilon - \beta \theta
\end{align*} \]

\[ \begin{align*}
\rho u_{tt} &= \nabla \cdot \sigma \\
\rho c_v \theta_t &= \nabla \cdot (\kappa \nabla \theta) - \beta T_0 \text{tr}(\varepsilon_t)
\end{align*} \]

- Volumetric strain rate drives energy transfer from mechanical to thermal domain
  - Irreversible diffusion \(\Rightarrow\) mechanical damping
  - Not often an important factor at the macro scale
  - Recognized source of damping in microresonators
- Zener: semi-analytical approximation for TED in beams
- We consider the fully coupled system
Nondimensionalization

\[
\begin{align*}
\sigma &= \hat{C}c - \xi \theta_1 \\
u_{tt} &= \nabla \cdot \sigma \\
\theta_t &= \eta \nabla^2 \theta - \text{tr}(\epsilon_t) \\
\end{align*}
\]

\[
\bar{\xi} := \left( \frac{\beta}{\rho c} \right)^2 \frac{T_0}{c_v} \text{ and } \eta := \frac{\kappa}{\rho c_v c L}
\]

<table>
<thead>
<tr>
<th>Length</th>
<th>~ ( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>~ ( L/c, \ where \ c = \sqrt{E/\rho} )</td>
</tr>
<tr>
<td>Temperature</td>
<td>~ ( T_0 \frac{\beta}{\rho c_v} )</td>
</tr>
</tbody>
</table>
Scaling analysis

\[
\begin{align*}
\sigma &= \hat{C}\epsilon - \xi \theta \\
u_{tt} &= \nabla \cdot \sigma \\
\theta_t &= \eta \nabla^2 \theta - \text{tr}(\epsilon_t)
\end{align*}
\]

\[\xi := \left( \frac{\beta}{\rho c} \right)^2 \frac{T_0}{c_v} \quad \text{and} \quad \eta := \frac{\kappa}{\rho c_v c L} \]

- Micron-scale poly-Si devices: \( \xi \) and \( \eta \) are \( \sim 10^{-4} \).
- Small \( \eta \) leads to thermal boundary layers
- Can linearize about \( \xi = 0 \) for a fast solver
- **Must** note scaling behavior to get accurate eigenvalues
Clamped SCS beam

- Mesh similar to before, but clamped at both ends
- Use full thermoelastic element (single-crystal Si)
- Given length scale and material properties, automatically compute other characteristic scales
- Compute with dimensionless model
Clamped SCS vibration

\[ Q = 8700 \text{ at } 87.7 \text{ MHz} \]

```matlab
nedopt.type = 'pert';
tedopt.cT = cT;
[V,w,Q] = tedmode(mesh,w0,1,tedopt);
```

- Compute thermoelastic modes using `tedmode`
- Two methods: full solve and perturbation-based
  - Full solve (default) is more expensive
  - Perturbation method only works for \( \zeta \) small
Checkerboard resonator

- Array of loosely coupled resonators
- Anchored at outside corners
- Excited at northwest corner
- Sensed at southeast corner
- Surfaces move only a few nanometers
Checkerboard mesh

- A more complex tied mesh example
- Hierarchical: “checker” and “beam” functions
- Can vary
  - Dimensions
  - Number of checkers
  - Linking strategy (corner-coupled or beam-coupled)
  - Drive and sense positions
- Define functions for drive and sense
  - Lua functions look like those passed to `set_bc`
    ```lua
    function drive_checker(x,y)
        ... return string and values ...
    end
    ```
  - MATLAB functions construct corresponding vectors
    ```matlab
    L = Mesh_get_sense_u(mesh, 'sense_checker');
    B = Mesh_get_sense_f(mesh, 'drive_checker');
    ```
Model reduction

Approximate frequency response near $\omega_0$ with a $k$-dimensional reduced model.

$$H(\omega) = L^T(K - \omega^2 M)B \approx L_k^T(K_k - \omega^2 M_k)B_k$$

```matlab
freq = linspace(wmin, wmax, n);
[Mk,Kk,Lk,Bk,Vk] = rom_arnoldi(M,K,L,B, k, w0);
H = compute_bode_mech(freq, Mk,Kk,Lk,Bk);
plot_bode(freq, H);
```
Bode plot

Transfer (dB)

Frequency (Hz)

Phase (degrees)

Frequency (Hz)
Interactive visualization

- Overview
- Introduction
- The Basics
- Anchor loss
- Thermoelastic damping
- Checkerboard and ROM
- Checkerboard resonator
- Checkerboard mesh
- Model reduction
- Bode plot
- Interactive visualization

Wrap-up
Current status

- Alpha version available with MATLAB and Lua front-ends
  - Already allows some interesting analyses
  - Ongoing verification work (E. Quévy)

- Pre/post-processing:
  - Language for parameterized mesh description
  - Mainly use tied meshes of mapped blocks
  - MATLAB graphics, some ability to export to OpenDX

- Elements:
  - Isotropic and anisotropic scalar, elastic, TED elements
  - Linear, quadratic, and cubic shapes supported
  - All elements allow PML mapping

- Analyses:
  - Modal analysis: standard, PML, and TED
  - Forced response and energy flux computation
  - Model reduction and fast Bode plots
  - Parameter sweeps on any of the above
Planned work

- Short term
  - 3D calculations
  - Intrinsic material loss models
  - Piezo models?
  - Studies of grain boundary effects in TED
  - Continued comparisons to disk resonators
  - Comparisons to flexural resonator data
  - User documentation
  - Finding a new name!

- Long term
  - Computer optimization of bandpass filter designs
  - Continued simulation/verification of actual devices

David Bindel, February 18, 2005
Conclusions

- HiQLab page: www.cs.berkeley.edu/~dbindel/hiqlab
- Questions or comments?